2006-05-03

Study on Genetic Algorithm Improvement and Application

Yao Zhou
Worcester Polytechnic Institute

STUDY ON GENETIC ALGORITHM IMPROVEMENT AND APPLICATION

by

Yao Zhou

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Manufacturing Engineering

by

_____

Yao Zhou

May 2006

APPROVED:

_____

Dr. Yiming (Kevin) Rong, Major Advisor,
Associate Director of Manufacturing and Materials Engineering,
Higgins Professor of Mechanical Engineering

ABSTRACT

Genetic Algorithms (GAs) are powerful tools to solve large scale design optimization problems. The research interests in GAs lie in both its theory and application. On one hand, various modifications have been made on early GAs to allow them to solve problems faster, more accurately and more reliably. On the other hand, GA is used to solve complicated design optimization problems in different applications.

The study in this thesis is both theoretical and applied in nature. On the theoretical side, an improved GA—Evolution Direction Guided GA (EDG-GA) is proposed based on the analysis of Schema Theory and Building Block Hypothesis. In addition, a method is developed to study the structure of GA solution space by characterizing interactions between genes. This method is further used to determine crossover points for selective crossover. On the application side, GA is applied to generate optimal tolerance assignment plans for a series of manufacturing processes. It is shown that the optimal tolerance assignment plan achieved by GA is better than that achieved by other optimization methods such as sensitivity analysis, given comparable computation time.

ACKNOWLEDGEMENTS

It gives me great pleasure to have the opportunity to thank the people who helped me during the thesis work and my studies at Worcester Polytechnic Institute.

I would like to express the deepest gratitude to Professor Yiming (Kevin) Rong, my advisor, for his help, guidance and encouragement in my thesis work and far beyond. Without his numerous suggestions and immense knowledge, the thesis would never have been completed. Without his open-mindedness, my experience of study here would not have been so rewarding.

I would like to thank Professor Richard Sisson and Professor Diran Apelian for their enthusiastic service on the committee. The research assistantship provided by the Center for Heat Treating Excellence (CHTE) at WPI is also acknowledged.

I am grateful for the help offered by Professor Karl Helmer from the Dept. of Biomedical Engineering, who gave me the eye-opening opportunity to try out projects on medical image analysis.

I appreciate the many inspiring discussions with Dr. Hui Song and the great help offered by all the other members in the Computer-aided Manufacturing Lab at WPI. I also thank the program secretary, Ms. Susan Milkman, for her help during the two years.

Special thanks should be given to my parents for their constant love and emotional support during my studies here, without which this work should not have been possible. I sincerely dedicate this thesis to my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

This chapter provides an introduction to Genetic Algorithm (GA) on both theory and application aspects. First, the fundamental problems of design optimization are addressed. Meta-heuristic methods as powerful solution tools are discussed in general. Then, GA is introduced, including its rationale, applications and research topics. Tolerance assignment is then introduced as a design optimization problem in which GA can be applied. Next, the objective of this thesis is defined. Finally, the organization of the thesis is given.

1.1 Design Optimization and Meta-Heuristic

1.1.1 Design Optimization

Design optimization is the issue of determining the set of design parameters that will optimize a given objective. Design optimization is of interest to many design problems, especially complicated problems. For example, when designing a composite material, one needs to determine the percentage content of each constituent, so that the best mechanical property of the composite can be achieved [1]. When designing a transportation system, one needs to determine the connections between numerous transportation nodes to ensure the system is robust and cost effective [2]. When scheduling manufacturing processes, one needs to decide when to use what

manufacturing resources to have the resources collaborate in a reliable and efficient manner [3].

Design optimization problems can be categorized in different ways. One way is to divide them into such two classes [4]: functional optimization and combinatorial optimization. In functional optimization, the objective function can usually be formulated as a continuous or piecewise continuous function of the design parameters. For example, the mechanical property of a composite material may be a continuous function of the percentage content of each constituent. On the other hand, in combinatorial optimization, the possible value of each parameter is discrete. Different combinations of such discrete parameters form finite number of "states" of the problem, which will affect the optimization objective in a certain way. The design of a transportation system is an example of combinatorial optimization problem.

Simple functional optimization problems can be solved through rigorous mathematical methods. However, when the functions are complicated, formal methods are inadequate. In such cases, the functional optimization problem can be discretized to transform into a combinatorial optimization problem, and then solved by methods for combinatorial problems. This thesis basically concerns combinatorial optimization problems.

Major difficulties in combinatorial problems are: 1) the solution space is too large for exhaustive search; 2) the relationship between the design parameters and the optimization objective has not been completely understood, and hence the problem cannot be solved through analytical methods.

1.1.2 Meta-Heuristic

Meta-heuristic algorithms are powerful and efficient tools to solve combinatorial optimization. Meta-heuristic algorithms are formally defined as iterative generation processes which guide a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space. Learning strategies are used to structure information in order to find efficiently near-optimal solutions [5].

Meta-heuristic are approximate algorithms rather than deterministic algorithms. Deterministic algorithms guarantee to find an optimal solution in bounded time for every finite size instance of a problem, but they often lead to computation times too high for practical purposes. As approximate algorithms, meta-heuristic sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time [6]. Many of the meta-heuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in meta-heuristic algorithms randomness is not used blindly but in an intelligent, biased form. Some widely adopted meta-heuristic methods include but not limited to Simulated Annealing(SA) [7], Tabu Search(TS) [8], Neural Networks(NN) [9], and Genetic Algorithms(GA) [10, 11].

1.2 Genetic Algorithm (GA)

1.2.1 Rationale

Genetic Algorithm (GA), first proposed by John Holland in 1975 [10], are a type of meta-heuristic search and optimization algorithms inspired by Darwin's principle of natural selection. The central idea of natural selection is the fittest survive. Through the process of natural selection, organisms adapt to optimize their chances for survival in a given environment. Random mutations occur to the genetic description of an organism, which is then passed on to its children. Should a mutation prove helpful, these children are more likely to survive to reproduce. Should it be harmful, these children are less likely to reproduce, so the bad trait will die with them [12].

In analogy, GA maintains a "population" of solution candidates for the given problem. Elements are drawn at random from this population and allowed to "reproduce", by combining some aspects of the two parent solutions. The key is that the probability that an element is chosen to reproduce is based on its "fitness", essentially an objective function related to the solution. Eventually, unfit elements die from the population, to be replaced by successful solution offspring [12].

In GA, solutions are parametrically represented in strings of code (e.g. binary). Fitness value is defined to evaluate solutions. The general procedures of a GA include: 1) Create a population of random individuals; 2) Evaluate each individual's fitness; 3) Select individuals to be parents; 4) Produce children; 5) Evaluate children; 6) Repeat steps 3 to 5 until a solution with satisfied fitness is found or some predetermined number of generations is met [13]. More details on GA procedures are further treated in Chapter 3.

1.2.2 Applications

Since the inception of GA, they have found applications in numerous areas. In the area of engineering design, Yao (1992) used GA to estimate parameters for nonlinear systems [14]; Joines (1996) applied GA to manufacturing cell design [15]; Gold (1998) introduced GA to kinematic design of turbine blade fixtures [16]. In the area of scheduling and planning, Timothy (1993) optimized sequencing problems using GA [17]; Davern (1994) designed the architecture for job shop scheduling with GA [18]. In the area of computer science, Rho (1995) used GA in distributed database design [19]. In the area of image processing, Tadikonda (1993) used GA to realize automated image segmentation and interpretation [20]; Huang (1998) designed detection strategies for face recognition with GA [21].

Due to the fact that GA is non-problem-specific, its application is not confined with the problems' physical background, and hence can be applied to many combinatorial optimization problems in different disciplines.

1.2.3 General Research Topics

The interest in understanding and promoting GA's performance motivated considerable theoretical research of GA. The ultimate goal is to be able to design efficient and robust GAs. To achieve this goal, however, two fundamental questions should be fully understood: 1) How do GAs work? 2) What types of problems are suitable for GAs

to solve [22]? Generally speaking, almost all the theoretical work in GA stems from these two fundamental questions.

To answer the first of the two questions, mathematical tools are adopted to model the evolution process and explain the improvement of solutions over generations. The first relatively rigorous model should be credited to John Holland [10]. He developed Schema Theorem to describe how certain pieces of code thrive or diminish depending on their fitness relative to the average. Although there are some criticisms against Schema Theorem, it still serves as the basis for many theoretical studies of GA. Some research related to Schema Theorem includes the modification of the theorem [23]. Another attempt is to model the GA process as a Markov process [24]. The Markov models aim at describing the convergence behavior of GA. It is more precise, but unfortunately, this model is usually very complicated and offers little practical guide to designing competent GAs. Based on the study of the mechanisms of GA, Goldberg proposed Building Block Hypothesis. He then decomposed the design of GA and provided several guidelines to the design of competent GAs [25]. From there, a series of GA was designed [26, 27, 28] with improved efficiency and/or robustness.

GA does not work well for all the problems. Thus, it is important to understand what type of problem GA is capable in solving, or alternatively what makes it difficult for GA. The central idea to address this question is the idea of epistasis. Simply put, epistasis refers to the interdependency between the parameters of a solution, which incurs nonlinearity and hence makes the problem hard for GA. Davidor proposed a method to measure epistasis [29]. Vose and Liepins showed that in principle epistasis in any

problem can be reduced through different encoding schemes [30]. However, for complicate problems, devising such a coding scheme can be a formidable task.

1.3 Computer-aided Tolerance Assignment

Tolerance management and quality control are key elements for industries to improve quality, reduce overall costs and retain market share. Both engineering designer and manufacturing planner are concerned about the effects of tolerances. Designers like tight design tolerances to assure functioning products. Manufacturers prefer loose process tolerances to make parts easier with less cost. Therefore tolerancing becomes a critical link between product design and process planning. However, the research on design tolerance and process tolerance has long been neglected despite its importance.

A production plan defines all setups and processes required to produce quality products from raw parts. It also specifies process tolerances to guide the manufacturing processes and ensure the design tolerance can be achieved. Consequently, following questions need to be answered.

1. How to determine whether all designed tolerances can be achieved upon given production plan and process tolerances?

2. How to generated optimal process tolerances for production plan from design tolerances to improve product quality, productivity, and save cost?

3. How to ensure all the tolerance requirements are met in the manufacturing practice?

Tolerance analysis is an important mean to study and answer these questions. It consists of tolerance stack-up analysis, tolerance assignment/optimization and quality control. Assuming all manufacturing errors are known, the tolerance stack-up analyzes the variations of finished workpiece and predicts whether all the design tolerance requirements are satisfied. Tolerance assignment finds a set of feasible process tolerances for all the setups and processes. The aim of optimization is to achieve optimal manufacturing tolerance assignment plan to minimize cost/cycle time while maintaining product quality. The tolerance stack-up and assignment analysis can help to develop quality control plan, as answer to the last question.

As the global competition driving industrial companies to pursue higher quality and lower cost, it is desired that even products developed in small volume can be manufactured with economic mass product mode, i.e., mass customization. This requires optimal production plan to be made rapidly according to the available manufacturing resources. Study and development of computer-aided tolerance analysis (CATA) are driven by the demand of industry and accommodated by the rapidly improving computer technology.

1.4 Thesis Objective

The effort of this thesis is in two fold. One is to enhance the performance of GA through theoretical analysis and development of the algorithms. The other is to apply GA to solve complicated design problems in manufacturing in order to lower cost.

The objectives of this thesis are:

1) To design an enhanced GA based on the analysis of Schema Theorem and Building Block Hypothesis in order to expedite the evolution process of GA;

2) To propose a method to study the solution space structure and characterize interactions between genes, and use such information to enhance GA by modifying the crossover mechanism;

3) To apply GA to facilitate tolerance assignment in manufacturing planning.

1.5 Thesis Organization

Chapter 2 reviews the research relevant to the work in this thesis, including competent GA decomposition, efficiency enhancement techniques, and computer-aided tolerancing.

Chapter 3 proposes an enhanced GA—Evolution Direction Guided-Genetic Algorithm (EDG-GA). Based on the analysis of Schema Theorem and Building Block Hypothesis, EDGGA is designed by properly modifying the simple Genetic Algorithm (SGA) with the intention of predicting fit chromosome.

Chapter 4 proposes a methodology to study the structure of solution space of GA-based problems. Specifically, the method addresses the interaction between genes, and exploits this information to implement selective crossover.

Chapter 5 formulates the tolerance assignment problem and demonstrates the solution of the problem with GA. Results are shown and analyzed.

Chapter 6 concludes the thesis, summarizing the finished studies and discussing possible future studies.

CHAPTER 2 REVIEW OF RELATED RESEARCH

This chapter reviews the research related to the work in this thesis, including competent GA design decomposition, GA efficiency enhancement techniques, and tolerance assignment.

2.1 Competent GA Design Decomposition

The goal of GA design is to design GAs that solve hard problems fast, accurately and reliably. GAs that meet these criteria are called “competent GA”. Centered with Holland’s notion of building blocks(BB) [10], Goldberg decomposed the problem of designing competent GAs and proposed the key elements therein [25].

1) Know that GAs process BB.

2) Identify GA-hard or BB-wise-hard problems.

3) Ensure adequate supply of raw BB.

4) Ensure increase of superior BB.

5) Know BB takeover and convergence time.

6) Make good decisions between competing BBs.

7) Identify BBs and mix them well.

Once we understand that GAs process building blocks, and that there are problems where BBs are hard to preserve and evolve, we treat BBs as a kind of material quantity that must be transported through space and time. First, we ensure that we have enough raw BBs to solve the problem at hand (supply question). We then make sure that selection bias is strong enough and that the innovation operator is safe enough to ensure that BB market share grows with time (Schema Theorem or market share question). Thereafter, we consider how long convergence takes on average (convergence time), and ensure that the pool of choices is sufficiently rich to permit good solutions (the decision question). Finally, we make sure that different building blocks come together on the same string through effective exchange (BB mixing).

Over the years, many competent GAs have been designed. Their mechanisms vary significantly, but in general, they have the abovementioned key elements of GA design. Some competent GAs include messy GA [26], Linkage Learning GA [27], and Bayesian Optimization GA [28], etc.

2.2 Efficiency Enhancement of GA

Goldberg categorized the efficiency enhancement techniques of GA into four broad classes: parallelization, hybridization, time continuation, and evaluation relaxation [25].

1) Parallelization: GAs are executed on several processors and the computational load is distributed among these processors [31]. This leads to significant speed-up when

solving large scale problems. Parallelization can be achieved through different ways. A simple way is to have part of the GA operations such as evaluation run simultaneously on multiple processors [32]. Another way is to create several subpopulations and have them evolve separately at the same time, while spreading good solutions across the subpopulations [33].

2) Hybridization: Local search methods or domain-specific knowledge are coupled with GA. GAs are powerful in global search. However, they are not as efficient as local search methods in reaching the optimum on micro-scale. Therefore, hybridization which incorporates local search methods into GA will facilitate local convergence. A common form of hybridization is to apply a local search operator to each member of the population after each generation in GA [34].

3) Time Continuation: The capabilities of both mutation and recombination are utilized to obtain a solution of as high quality as possible with a given limited computational resource [35]. Time continuation exploits the tradeoff between the search for solutions with a large population and a single convergence epoch or using a small population with multiple convergence epochs.

4) Evaluation Relaxation: An accurate, but computationally expensive fitness evaluation is replaced with a less accurate, but computationally inexpensive fitness estimate. The low-cost, less-accurate fitness estimate can either be 1) exogenous, as in the case of surrogate (or approximate) fitness functions [36], where external means can be used to develop the fitness estimate; or 2) endogenous, as in the case of fitness

inheritance [37] where the fitness estimate is computed internally and is based on parental fitness.

2.2 Tolerance Assignment

Operational tolerance synthesis is an important area where the product designer and process planner often need to work closely together. Despite the intensive studies in tolerancing, this area has been neglected by most researchers. This section reviews papers on some closely related issues, e.g. the assembly tolerance synthesis/allocation, manufacturing cost models, and application of genetic algorithm in tolerancing.

2.2.1 Assembly Tolerance Synthesis/Allocation

Most of the established tolerance synthesis methods are focusing on assembly processes, allocating the assembly functional tolerance to the individual workpiece tolerance to ensure that all assembly requirements are met [38]. No existing technique has been found by the authors that generates process and locator tolerance requirements for production plan.

A variety of techniques have been employed to allocate assembly tolerances. Among them, integer programming for tolerance-cost optimization [39], rule-based approach [40], feature-based approach [41], knowledge-based approach [42], and

statistical methods [43], artificial intelligence [44] have been used to optimize tolerance allocation.

2.2.2 Manufacturing Cost Models

One of the ultimate goals of an enterprise is to make profit. Hence, every company has been struggling to reduce cost, which can be done more effectively at the design and planning stage rather than manufacturing stage. It has been shown that about 70% of production cost is determined at the early design phase [45].

Manufacturing cost modeling at design stage has been investigated for many years and used as one of the major criteria, if not the only, for optimization of production planning. There are numerous facets in cost models. One way is to interpret the manufacturing cost as summation of processing cost, inspection cost, rework/scrap cost, and external failure cost [46]. The processing cost can then be decomposed into machine cost, tool cost, material cost, setup cost, overhead cost, energy cost, etc [47]. All terms can be further formulated if adequate information on process characteristics is known. This method gives detail analysis on each factor that contributes to final cost. However, each term normally involves assumption-orientated undetermined terms, empirical/semi-empirical formulation, and/or production line data that may even not available all the time, which made it difficult to be widely implemented. The other method used to estimate production cost is feature based modeling. Instead of collecting all detail process information, this method directly link the manufacturing cost with features [48]. The assumption behind this method is that the company should be able to produce a quality

feature at competitive or prevailing rate. This rate is determined by the feature type and relationships between features. Some researchers adopted this method for assembly product design and evaluate the cost at feature level, component level, and assembly level [49]. Nonetheless, it is not commonly employed in production planning due to the lack of compliance with industrial standards.

2.2.3 Application of GA in Tolerancing

As stated earlier, genetic algorithm is one of the techniques that have been used for optimal tolerance synthesis/allocation. Genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. It is an iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges. During each generation the structures in the current population are rated for their effectiveness as solutions, and on the basis of these evaluations, a new population of candidate structures is formed using specific ‘genetic operators’ such as reproduction, crossover, and mutation. This search algorithm is good for system with unknown or implicit function, unlimited or very large searching space.

Statistic tolerancing, especially the developed Monte Carlo simulation based tolerance stack up analysis does not provide explicit relationship between the stack up results and the input process/locator tolerances. Furthermore, a multi-setup production line normally consists of dozens even hundreds of processes and each process can be set at one of several tolerance levels. Every combination of those process/locator tolerances is one candidate for tolerance synthesis plan. Evidently, the search space increases

exponentially with the number of processes. With this understanding, several researchers have applied genetic algorithm in statistic tolerancing [43].

2.4 Summary

Although various approaches have been proposed to design competent GA, making it work faster, more accurately and more reliably, space still exists in enhancing the GA's performance by exploiting the real-time information of the population during evolution process and creating guided bias. Also, studying the structure of GA solution space will help design more efficient GA operators.

On the application aspect, GA can be used to solve complicated design problems in manufacturing, such as the tolerance assignment problem that involves multiple setups and multiple processes.

CHAPTER 3 EVOLUTION DIRECTION GUIDED GENETIC ALGORITHM

This chapter presents an improved Genetic Algorithm—Evolution Direction Guided-Genetic Algorithm (EDG-GA). By predicting genes that are potentially fit and directing the evolution process towards potentially fit directions, EDG-GA attempts to reduce the number of iterations before satisfactory/optimal solutions occur in the population. This is achieved heuristically by extracting real-time information from the evolution process and replacing the worst chromosome in the current generation with a newly-constructed chromosome whose alleles are determined adaptively based on such information.

First, the basic procedures of the Simple Genetic Algorithm (SGA) were introduced. Then, the theoretical foundations of GAs—Schema Theorem and Building Block Hypothesis are introduced and further analyzed in detail. Next, the enhanced algorithm EDG-GA was introduced based on the analysis. Finally, some limitations of EDG-GA are discussed, which will be further studied in next chapter.

3.1 Simple Genetic Algorithm (SGA)

Many different GAs have been developed in the same light of natural selection and evolution. They may have different operators, however, most GAs are derived from a

common “prototype”—Simple Genetic Algorithm (SGA), which has only the basic reproduction operators.

An SGA consists of the following procedures: encoding, defining objective function, initializing population and then an iterative process including evaluation and reproduction (selection, crossover and mutation). (Figure 3.1)

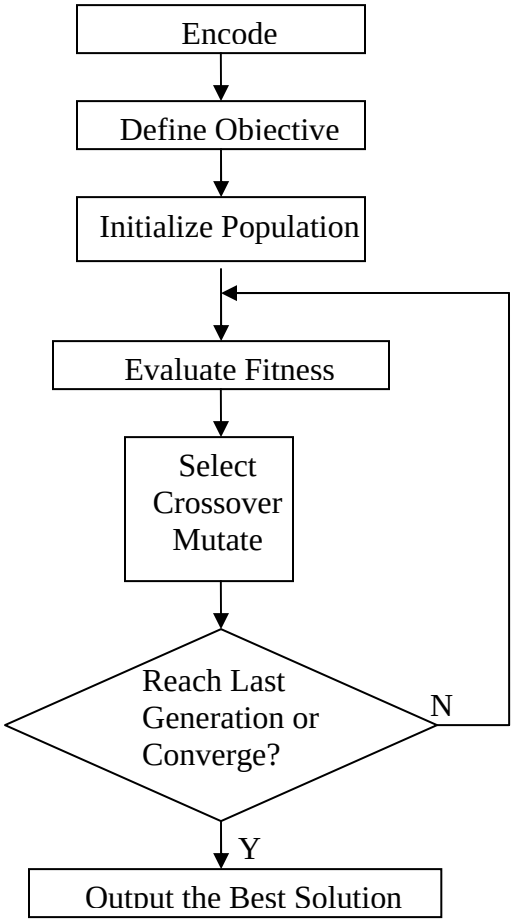

Figure 3.1 Flowchart of the procedure of SGA

Encoding means representing the potential solutions which contain a set of decision variables (e.g. a set of dimensions of a workpiece) as strings of codes. The decision variables can be coded in various formats, such as binary code, letter, real

number, etc. However, the classical encoding usually uses binary code. The coded variables are called genes, and the actual values of genes (e.g. either 0 or 1) are called alleles. Each solution is constructed by linking the pieces of genes into a finite-length string (usually with fixed length), called a chromosome. For example, a chromosome can have a form of X=1001100111. GAs work with the coding itself rather than the decision variables behind the coding.

Once the solutions have been represented, we need to be able to tell how good a solution is. Thus, we need to define the objective function. As GAs work purely with the coding and do not care about the physical aspects of the problem, one advantage of GA is that it can handle the optimization problems where close-form relationship between decision variables and fitness function is unknown or too complicated to formulate. The objective function value of an individual solution can be achieved through either a look-up table or computer simulation. This step establishes a one-one mapping between the variables and the function, with their relationship in a black box. The goal can be either to maximize or minimize the objective function.

After the solutions are represented and the objective function is defined, we now can proceed to the evolution process. First, an initial population of coded solutions or chromosome is created randomly or through pre-learnt knowledge. Then, the population undergoes an iterative process of evaluation and reproduction. Evaluation assigns each chromosome with a fitness value according to the objective function. After evaluation, chromosomes of the new generation are created by applying reproduction operators on the old generation. There are three basic reproduction operators: selection, crossover and mutation. The newly created chromosomes will again be evaluated. This iteration

continues until the population of solution converges to an optimal solution or other terminating criteria are satisfied.

Selection operator creates different number of copies of each chromosome for later operations. Selection procedure adopts the rule of “survival of the best” by allocating more quotas of copies to fitter solutions. There are different selection methods. The most commonly used method in SGA is the proportional selection, which means the number of copies of each chromosome is proportional to its relative fitness value: The higher the fitness value, the more the copies. Then the adjusted population will replace the original population for later operations. By doing this, it essentially creates a biased chance of occurrence of the chromosomes, favoring the fit ones in the population.

Crossover operator exchanges the pieces of genes between chromosomes. First, the population after selection is divided into groups, each of which contains two chromosomes. Then the two chromosomes (parents) in each group swap segments of their codes with each other to create two new chromosomes (children). Then the children will replace their parents in the population. Through crossover, it introduces new chromosomes to the population, and hence the possibility of having fitter chromosomes. This cannot be achieved by selection. There are many crossover schemes, such as one-point crossover, multi-point crossover and uniform crossover. Figure 3.2 shows how two-point crossover works.

Mutation operation alters individual alleles at random locations of random chromosomes at a very probability, e.g. from 0 to 1 or 1 to 0 (Figure 3.3). Mutation is a rather random operation. It might create a better or worse chromosome, which will either

thrive or diminish through next selection. Since the goal is to find optimum, it does hurt if the population has a single bad solution temporarily. On the other hand, however, it will be very helpful if a good solution is generated through mutation.

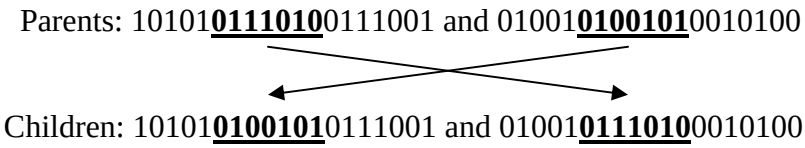

Figure 3.2 Two-point crossover

Figure 3.3 Mutation

3.2 Schema Theorem and Building Block Hypothesis

Schema Theorem and Building Block Hypothesis are the foundations of GAs' evolution mechanisms.

To address one of the two fundamental questions in GA, that is how GA works, many attempts have been made to explain the evolution mechanisms of GA. Schema Theorem was the first relatively rigorous explanation of such mechanisms. It describes in mathematical forms how selection, crossover and mutation operators work to prosper fit schemata and suppress unfit ones. "Schema" refers to a subset of the solution space in which all chromosomes share a particular set of defined alleles. For example,

chromosomes (1,0,0,1,0) and (1,1,0,1,1) share the schema (1,*,0,1,*), or equally, chromosomes (1,0,0,1,0) and (1,1,0,1,1) are both instances of the schema (1,*,0,1,*), where * means this digit is not specified.

Schema theorem gives the possibility for a certain schema to survive into the next generation given that it appears in the current generation. The classic form of expression of Schema Theorem is the following inequality:

$$E[N(S,t+1)] \geq \{1-\chi\frac{l(S)}{l-1}-\mu k(S)\}r(S,t)N(S,t)$$

N(S,t): number of instances of schema S at generation t.

E[N(S,t+1)]: expected number of instances of schema S at generation t+1.

l: length of a chromosome in terms of the number of digits.

l(S): length of schema S, defined as the distance in number of digits between the first and last specified digit in schema S. e.g.: *l*[(*,1,*,0,*)]=2.

k(S): order of schema S, defined as the number of specified digits. e.g.: k[(1,*,0,1,*)]=3.

χ : crossover rate.

μ : mutation rate.

r(S,t): fitness ratio, where $r(S,t) = f(S,t)/\overline{f}(t)$

$$f(S,t) = \frac{\sum_{x\in S\cap P(t)} f(x)}{|S\cap P(t)|} \qquad \overline{f}(t) = \sum_{x\in P(t)} f(x)/|P(t)|$$

f(S,t): average fitness of schema S at generation t.

$\overline{f}(t)$: average fitness of population at generation t.

x: individual strings.

|P(t)|: size of population P(t).

$|S \cap P(t)|$: size of the set which consists of all the strings from the population P(t) that are instances of schema S.

Here, the average fitness of a schema S is assumed to be the average fitness of all instances of schema S in the population P(t). Apparently, the larger population size, the less error this assumption will generate.

The right hand side of the inequality is actually made up of three parts, corresponding to selection, crossover and mutation, respectively. $r(S,t)N(S,t)$ gives the expected number of instances of schema S after fitness-proportional selection operation; while $\chi \frac{l(S)}{l-1}$ and $\mu k(S)$ give the probability that schema S is destructed due to one-point crossover and mutation, respectively. (Detailed proof can be found elsewhere [10].) So the expectation of the number of schema S in the next generation after all three operations (selection, crossover and mutation) is $\{1-\chi \frac{l(S)}{l-1}-\mu k(S)\}r(S,t)N(S,t)$.

However, one possibility is overlooked: although a schema may be destructed through crossover and mutation, the same schema may also be reconstructed from instances of other schemata through crossover and mutation. Consider one-point cross over, for instance, (*,*,1,0,*) may be destructed if the crossover point is between the third and forth digits. However, if chromosomes (1,1,1,1,0) and (1,1,0,0,0) exist in the population and happen to be grouped together to perform crossover, furthermore, the crossover point happens to be between the third and forth digits, then the a new chromosome (1,1,1,0,0) as an instance of (*,*,1,0,*) is reconstructed. So, without

considering the reconstruction effect, the right hand side in the formula $\{1-\chi\frac{l(S)}{l-1}-\mu k(S)\}r(S,t)N(S,t)$ is actually a lower bound of the expected number of schema S in the next generation $E[N(S,t+1)]$.

Building block refers to schemata with relatively short defining lengths and with fitness values above average. According to Schema Theorem, the number of these building blocks in the population will grow exponentially over generations. However, where do those good final solutions come from? Building Block Hypothesis states that through selection, crossover and mutation, good chromosomes are eventually achieved by connecting many fragments of fit building blocks. In other words, the hypothesis assumes the relationship between the fitness of the building blocks and that of the chromosome. Although there is no rigorous proof for Building Block Hypothesis, it is valid in most cases, as has been demonstrated through numerous applications.

3.3 Analysis of Schema Theorem and Building Block Hypothesis

According to Building Block Hypothesis, if we want to find good solutions or chromosomes, we should first look for their building blocks, i.e. shorter fit schemata. Then we connect many of such schemata into a fit chromosome.

Now, we examine the order-1 schemata, where only one digit is specified, such as (*,*,1,*,*) and (*,0,*,*,*), etc. For these schemata, the order k(S)=1, and the length l(S)=0. Then the Schema Theorem will reduce from its original form

$$E[N(S,t+1)] \geq \{1-\chi\frac{l(S)}{l-1}-\mu k(S)\}r(S,t)N(S,t) \text{ to } E[N(S,t+1)] \geq \{1-\mu\}r(S,t)N(S,t),$$

So, for order-1 schemata, the number of instances of a certain schema to appear in next generation depends upon: its number of instances in current generation, the fitness of the schema relative to the average and mutation rate. We further rewrite the reduced formula by rearranging terms on each side, and we get:

$$r(S,t) \leq \frac{E[N(S,t+1)]}{N(S,t)\{1-\mu\}}$$

For order-1 schemata, reconstruction of schemata takes place only through mutation, not through crossover. Since mutation rate is usually low, reconstruction effect can be neglected, so the above inequality can be approximated by the formula:

$$r(S,t) \approx \frac{E[N(S,t+1)]}{N(S,t)\{1-\mu\}}$$

Suppose the actual number of instances of a schema does not deviate significantly from the expected one (and it is true, otherwise the Schema Theorem will be useless). Then we can get:

$$r(S,t) \approx \frac{N(S,t+1)}{N(S,t)\{1-\mu\}}$$

Thus, if we already know the numbers of instances of a schema in two consecutive generations as well as mutation rate μ, we can infer the relative fitness of this schema. Recall that in the beginning of this section, we were aiming at finding shorter fit schemata. Specifically in this case, we want to find fit order-1 schemata, that is, order-1

schemata with high $r(S,t)$. Since mutation rate μ is a constant, we can determine such schemata by picking the ones with its number of instances having the largest increment by percentage between two consecutive generations.

By far, we have been able to identify all the potentially fit order-1 schemata at every digit or locus of the chromosome, which are, say, (1,*,*,*,*), (*,0,*,*,*), (*,*,1,*,*), (*,*,*,1,*), (*,*,*,*,0). According to the Building Block Hypothesis, it is reasonable to project that chromosome (1,0,1,1,0), which is composed by all the fit order-1 schemata, has a higher probability to be fit than other chromosomes.

Such projection exploits the information of the evolution process by noticing the change in bit-wise configurations of chromosomes between generations. This information is heuristic, meaning it might not hold true all the time. However, it should work in most cases, provided the Building Block Hypothesis is valid, which is evident in many applications. Also, this information itself might change over the generations as evolution process continues.

It should be beneficial that in each generation, we add one such projected chromosome to the population and have it replace the current worst chromosome. If the new chromosome is indeed good, it will soon thrive and provide further fit schemata to the population through crossover, and hence facilitate the evolution towards the optimum. In case the new chromosome is a bad bet, it will die soon in the coming selection cycle, so it won't hurt the overall fitness of the population. Moreover, as evolution goes on, the information based on which to project the fit chromosome is continuously updated. So the projected fit chromosome is always tailored to the recent generations.

It is noteworthy that since the key is to find the best, one solution that scores 10 is better than ten solutions that score 8. In this sense, projecting a single fit chromosome is better than some other methods that attempt to improve the average fitness of the entire population without seeking for individuals that stands out.

3.4 Evolution Direction Guided-Genetic Algorithm

EDG-GA is developed according to the analysis presented in the last section,. After each generation is formed by reproduction and before it moves on to evaluation in the next loop, EDG-GA adds an additional operation on top of original SGA. This new operation is implemented as follows: (Suppose we are at generation t)

a) At locus i ($i \in [0, L-1]$, L is the length of an individual), count the number of times allele a(i) ($a(i) \in$ {all possible alleles at locus i}) appears in the population at generation t-1 and t, noted as $N_{t-1}[i,a(i)]$ and $N_t[i,a(i)]$ respectively.

b) Generate such an chromosome that at each locus, the allele is the one with the maximum $N_t[i,a(i)]/N_{t-1}[i,a(i)]$ among all possible alleles.

c) Evaluate all chromosomes including the newly generated one.

d) Replace the worst chromosome with this new individual.

e) Continue to selection and so on.

Example:

Suppose our individual space is $\{x,y,z\}^3$. At generation t-1 and t, we have the following populations respectively:

Generation t-1: (y,x,z) (y,y,y) (z,z,y) (x,x,z) (x,y,y) (y,x,z) (x,z,z) (y,z,x)

Generation t: (x,y,z) (y,z,y) (x,x,y) (z,x,z) (z,y,x) (y,y,x) (z,x,y) (y,y,z)

For generation t-1, at locus 0, x appears 3 times, y appears 4 times, and z appears once. So $N_{t-1}[0,x]=3$, $N_{t-1}[0,y]=4$, $N_{t-1}[0,z]=1$. Likewise, all other $N_{t-1}[i,a(i)]$ and $N_t[i,a(i)]$ as well as $N_t[i,a(i)]/N_{t-1}[i,a(i)]$ can be calculated. They are listed in the following table:

	N_{t-1}			N_t			N_t/N_{t-1}		
a(i) / i	x	y	z	x	y	z	x	y	z
0	3	4	1	2	3	3	0.67	0.75	3
1	3	2	3	3	4	1	1	2	0.33
2	1	3	4	2	3	3	2	1	0.75

Table 3.1 Parameters for determining the projected chromosome

At loci 0, 1 and 2, the alleles with the largest $N_t[i,a(i)]/N_{t-1}[i,a(i)]$ are z, y and x respectively. So we generate a new individual (z,y,x). After evaluating all the individuals (including the new one) in generation t, replace the worst one in the population with the new one. The rest operations remain the same.

However, there are circumstances where at some loci, more than one alleles have the maximum value of $N_t[i,a(i)]/N_{t-1}[i,a(i)]$. In this case, an allele will be chosen randomly from them to fill these loci of the new chromosome. Also, it is possible that at

some loci, the number of occurrence of every allele did not change from last generation. In this case, the alleles of these loci of the newly generated chromosome remain the same as the one in last generation.

This operation is repeated in every generation. By doing so, we are continuously predicting the potentially fit chromosome based on the real time bit-wise information extracted from current generation. It is an adaptive process. Adding such a chromosome into the population generally leads the evolution to a fit direction.

3.5 Test Function

We use a 10-bit One-max function [50] to test the modified algorithm EDG-GA. In Onemax problem, the chromosomes are in binary codes. The value of fitness function of a chromosome equals the number of "1"s the chromosome has. For example, the 10-bit chromosome (0 1 0 0 1 1 1 0 1 1) has six "1"s, so its fitness value is 6. Obviously, the more "1"s a chromosome has, the more fit it is. The global optimum for a 10-bit One-max problem is the chromosome with all its genes being "1", and its fitness value is 10. The number of generations before optimum is reached is one of the measures to test the effectiveness of an algorithm on this type of problem.

A population of 20 chromosomes was generated at random. Proportional selection and one-point crossover were adopted. Mutation was turned down. SGA and EDG-GA were applied to solving the problem respectively. When the problem was solved with SGA, the first occurrence of the optimum happened in 16^{th} generation. In comparison,

when solving the problem with EDG-GA, the first occurrence of the optimum happened in 5^{th} generation. It was also noted that the optimum achieved in EDG-GA happened to be the newly generated predictive chromosome in 5^{th} generation. These results proved that with the introduction of the predictive chromosome in each generation, EDG-GA is effective in enhancing the computation efficiency for problems of the same type as One-max.

3.6 Summary

Based on the analysis of Schema Theorem and Building Block Hypothesis, a new GA — EDG-GA is proposed to facilitate the evolution process by projecting potentially fit chromosome using real-time inter-generational information. The new operator in EDG-GA works well for the problems in which the fitness of a chromosome and that of genes have a close-to-linear relationship. In this case, the fitness of a chromosome is contributed by the individual genes in an additive manner.

However, this is not always the case. There are problems with very complicated solution space structures. In these problems, the chromosomes and their genes appear to have highly nonlinear relationship. “Nonlinear” here means there is an ineligible degree of interactions between genes, so the fitness of a chromosome is not determined by individual genes separately, but rather by the collective effect of multiple genes. Thus, simply connecting individual genes that are fit does not necessarily produce fit chromosome. This type of problem is generally hard for any kinds of GAs, and is especially hard for EDG-GA. Since the newly added chromosome in EDG-GA is merely

an immediate connection of fit individual genes and neglects gene interactions, so the chances of false prediction could be escalated.

To be able to understand such gene interactions, a method to study the structure of the solution space of a problem is called for.

CHAPTER 4 STUDY THE STRUCTURE OF SOLUTION SPACE

This chapter proposes a systematic methodology to study the structure of solution space of a problem, and the application of such knowledge to design more efficient crossover operator. As mentioned in the end of last chapter, gene interactions bring nonlinearity to the problem, which increases the difficulty for GA solution. By depicting relationship between genes, the structure of solution space can in some degree be revealed.

First, the Gene-Chromosome Correlation (GCC) Function is defined to characterize the contribution of individual genes to the fitness of the chromosome. Then the dependency of gene *i* on gene *j* is achieved by comparing the GCCs of gene *i* when gene *j* has different values. When the degrees of dependency between all genes have been determined, an overall profile of the relationship between the genes can be represented using matrix. This is helpful to the understanding of the structure solution space. Furthermore, influential genes can be identified, and clusters of interdependent genes can be separated. In this study, binary coding is assumed. Also, large population is assumed to reduce statistical error.

4.1 Gene-Chromosome Correlation Function

To characterize the overall contribution of an individual gene at a certain locus to the fitness of the chromosome, we define the Gene-Chromosome Correlation (GCC) Function at generation t as follows:

$$GCC(i) = \frac{\sum_{X \in P(X)} [a_i(X) - \overline{a_i}] \cdot [f(X) - \overline{f}]}{\sigma[a_i(X)] \cdot \sigma[f(X)]}$$

$$= \frac{\sum_{X \in P(X)} [a_i(X) - \overline{a}] \cdot [f(X) - \overline{f}]}{\sqrt{\sum_{X \in P(X)} [a_i(X) - \overline{a}]^2} \sqrt{\sum_{X \in P(X)} [f(X) - \overline{f}]^2}}$$

GCC(i): correlation between the *i*th gene and the fitness of chromosome

$X \in P(X)$: chromosome X that belongs to population P(X)

$a_i(X)$: value of gene (allele) at locus *i* in chromosome X, either 0 or 1

$\overline{a_i}$: average allele at locus *i*, a value between 0 and 1.

$\overline{a_i} = \frac{\sum_{X \in P(X)} a_i(X)}{|P(X)|}$, $|P(X)|$: the size of population P(X)

$f(X)$: fitness function of chromosome X

$\overline{f}$: average fitness of all the chromosomes in population P(X), $\overline{f} = \frac{\sum_{X \in P(X)} f(X)}{|P(X)|}$

$\sigma[a_i(X)]$: standard deviation of $a_i(X)$

$\sigma[f(X)]$: standard deviation of $f(X)$

All terms are based on the same generation t. From the properties of correlation, we know *GCC(i)* is a value between -1 and 1. If *GCC(i)*>0, we can infer that at locus *i*, allele 1 is relatively more helpful for the fitness of chromosome comparing to allele 0 at the same locus. On the contrary, if *GCC(i)*<0, allele 0 is relatively more helpful at this locus. *GCC(i)*=0 indicates absolutely no relevance between this gene and the chromosome, while this is usually unlikely to happen. The larger the *absolute* value of *GCC(i)*, the larger the relevance of gene at locus *i* to the chromosome fitness. Thus, the Gene-Chromosome Correlation Function is able to characterize the contribution of genes at every locus to the chromosome fitness. GCC tells whether the genes are positive relevant (allele 1 is more desirable) or negative relevant (allele 0 is more desirable) to the fitness, and how big the relevance is.

4.2 Inter-Genic Dependency

Now, we modify the definition of GCC by taking into account the gene at one other locus. Suppose we look at locus $j\,(j \neq i)$. We separate the population into two sub-populations based on the value of gene at locus *j*. One sub-population contains all the chromosomes that have allele 1 at locus *j*, and is denoted as $P(X)|_{a(j)=1}$. Obviously, the other sub-population contains all those with allele 0 at locus j, and is denoted as $P(X)|_{a(j)=0}$. We now apply GCC function to each of these two sub-populations separately. The two GCC functions are denoted as $GCC(i)|_{a(j)=1}$ and $GCC(i)|_{a(j)=0}$ correspondingly. Note that all variables used to calculate each GCC function,

including $a_i(X)$, $\overline{a_i}$, $f(X)$, $\overline{f}$, $\sigma[a_i(X)]$, $\sigma[f(X)]$, are based on the two separate sub-populations $P(X)|_{a(j)=1}$ and $P(X)|_{a(j)=0}$ respectively.

If the contribution of gene at locus *i* to the chromosome, either in amount or polarity or both, is dependent on the value of gene at locus *j*, then there should be difference between the values of $GCC(i)|_{a(j)=1}$ and $GCC(i)|_{a(j)=0}$, and vice versa. Therefore, we define the dependency of *i*th gene on *j*th gene as

$$R(i, j) = \sigma[GCC(i)]|_{a(j)} / 2 = \sqrt{\frac{1}{|P(X)|}\sum_{a(j)}[GCC(i)\Big|_{a(j)} - \overline{GCC}]^2} / 2 \quad (j \neq i)$$

$\sigma[GCC(i)]|_{a(j)}$ is the standard deviation of GCC(i) when *j*th gene is set to different values. *R(i, j)* is then again a value between -1 and 1. The absolute value of *R(i, j)* indicates the degree of dependency: the greater the absolute value of *R(i, j)*, the greater the dependency. The sign of *R(i, j)* indicates the type of dependency, either positive or negative.

4.3 Dependency Matrix

Once the dependency values between all genes are calculated, a dependency matrix **R** containing all these dependency values can be constructed. Let the dependency value *R(i, j)* be the element in the *i*th row and *j*th column of the matrix **R**. The diagonal elements of the matrix are set to be 1, since a gene is always totally dependent on itself.

Hence, if there are l genes in the chromosome, the dependency matrix **R** has the following form:

$$\begin{array}{c} \\ Gene1 \\ Gene2 \\ \vdots \\ Genel \end{array} \begin{array}{c} \begin{array}{cccc} Gene1 & Gene2 & \cdots & Genel \end{array} \\ \begin{bmatrix} 1 & R(1,2) & \cdots & R(1,l) \\ R(2,1) & 1 & \cdots & R(2,l) \\ \vdots & \vdots & \vdots & \vdots \\ R(l,1) & R(l,2) & \cdots & 1 \end{bmatrix} \end{array}$$

Figure 4.1 Dependency matrix **R**

For each generation, we will have a dependency matrix **R**. Suppose the current generation is generation t, then the dependence matrix is denoted as $\mathbf{R_t}$. To increase the reliability of the dependency values, we make use of the knowledge of all generation up to present by averaging the elements in **R** over generations to get average dependency matrix $\overline{R}_t$

$$\overline{R}_t(i,j) = \frac{\sum_{s=0}^{t} R_s(i,j)}{t}$$

This matrix is continuously updated over generations. With this matrix, we can have a complete knowledge of how much each gene is related to others, and in what way (positive or negative). Each row tells how much a certain gene is dependent on all the other genes, and each column tells how much all other genes on dependent on this gene. This matrix accounts for the nonlinearity imported by the interactions between genes. By

doing so, this matrix essentially represents information about the structure of the solution space of the problem.

Finally, we apply a threshold g ($0<g<1$) on all the elements in the matrix so that a new matrix $\mathbf{M_t}$ containing only 0 and 1 is constructed.

$$M_t(i,j) = \begin{cases} 1 & if \quad \left|\overline{R}_t(i,j)\right| > g \\ 0 & if \quad \left|\overline{R}_t(i,j)\right| < g \end{cases}$$

where $M_t(i, j)$=1 indicates a strong interaction, and $M_t(i, j)$=0 indicates a weak interaction. By doing this, we are able to separate the strong gene interactions from the weak ones.

4.4 Building Block Identification

One major hindrance to the efficiency of GAs is the destruction of fit Building Blocks. To address this problem, we recall the Schema Theorem Inequity:

$$E[N(S,t+1)] \geq \{1-\chi\frac{l(S)}{l-1}-\mu k(S)\}r(S,t)N(S,t)$$

(Refer to Chapter 3 for explanation of notations)

The term $\chi\frac{l(S)}{l-1}$ on the right hand side represents the probability that a schema is destructed through crossover *if the locations of crossover points are random.* Aside from the crossover rate χ and chromosome length l which are constants, this probability is

determined by the length of the schema $l(S)$. The longer the schema, the more likely it will be destructed through crossover.

According to Building Blocks Hypothesis, fit chromosomes are made up by fit building blocks. These Building blocks are essentially schemata of different lengths that *can effectively influence the chromosome fitness*. The lengths $l(S)$ of building blocks depend on the problem. For nonlinear problems where gene interactions are strong, the chromosome fitness is influenced by groups of genes collectively rather than by individual genes separately. So, the building blocks usually have larger size, which in turn means greater defining lengths $l(S)$, and hence they are easier to be destructed.

In summary of the above analysis, for nonlinear problems, if the locations of crossover points are random, these building blocks are more likely to be destructed. To preserve fit building blocks, the goal is to design a crossover scheme that has selective crossover points, not random.

We want to select crossover points that lie between different building blocks rather than within building blocks. The task is then to identify these building blocks. The way to do it is to group all the genes into different clusters such that interactions only exist between genes within the same cluster but not between genes of different clusters. Recall that the matrix $\mathbf{M_t}$ defined earlier records all the strong interactions between genes in 0 and 1. By a series of matrix manipulation [51], we can transform $\mathbf{M_t}$ from its original form as in Figure 4.2 into a clustered form as in Figure 4.3. In the case shown in the figures, the genes are then clustered as [(B, D, G), (A, C, E, H), F]. Thus, genes B, D, G

are never separated during crossover, and neither are genes A, C, E, H. As a result, the building blocks are identified and preserved, and the efficiency of GA can be enhanced.

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
A	1		1		1			1
B		1		1			1	
C	1		1		1			1
D		1		1			1	
E	1		1		1			1
F						1		
G		1		1			1	
H	1		1		1			1

Figure 4.2 $\mathbf{M_t}$ in its original form

	B	*D*	*G*	*A*	*C*	*E*	*H*	*F*
B	1	1	1					
D	1	1	1					
G	1	1	1					
A				1	1	1	1	
C				1	1	1	1	
E				1	1	1	1	
H				1	1	1	1	
F								1

Figure 4.3 $\mathbf{M_t}$ in its clustered form

4.5 Summary

Gene-Chromosome Correlation Function is defined to characterize the contribution of each individual gene to the chromosome fitness. Based on this function, dependency matrix is defined to address the interactions between genes. Dependency matrix represents the structure of solution space by recording the degree of interactions between every two genes. The modified dependency matrix is further used to design efficient crossover operator that is able to identify and preserve building blocks.

CHAPTER 5 COMPUTER-AIDED TOLERANCE ASSIGNMENT USING GENETIC ALGORITHM

This chapter presents the application of genetic algorithm in computer aided tolerance assignment. First, the background of tolerance assignment is briefly treated. Then, tolerance assignment is formulated into a constraint optimization problem. Cost model is discussed in preparation for the design of objective function in GA. Next, the procedures of tolerance assignment using GA are presented. Finally, a specific case is shown. Computer simulation is implemented and the results are compared with those achieved from other methods.

5.1 Background

There are two sets of tolerances: design tolerances and operational tolerances. Design tolerances refer to the dimensional errors specified for the features of a part. They are determined by product designers based on the part's functional and assembly requirements. Each feature is machined through one or more manufacturing processes. The errors incurred in these processes are called operational tolerances. Operational tolerances in all the processes will stack up to generate the final summed errors. These summed errors have to be smaller than design tolerances. The tolerance stack-up is shown in Figure 5.1. Thus, the tighter the design tolerances are specified, the tighter the operational tolerances have to be. To guarantee design specifications are satisfied,

designers want tolerances to be tight. However, to minimize manufacturing cost, manufacturers usually want tolerances to be loose.

Tolerance assignment is the task to determine operational tolerances. The goal of tolerance assignment is to determine the operational tolerances that will minimize overall manufacturing cost provided that the stack-up of operational tolerances does not exceed design tolerances.

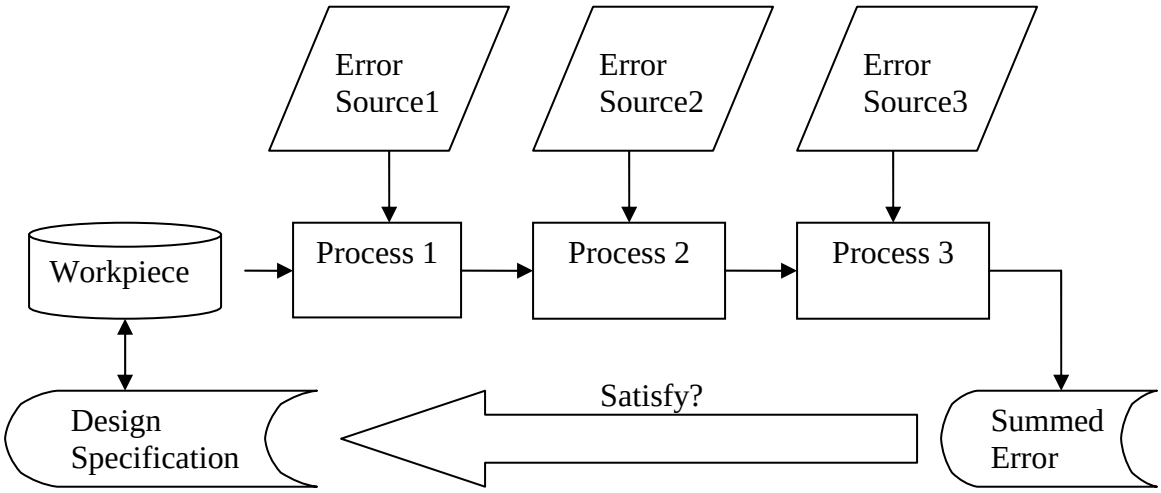

Figure 5.1 Tolerance stack-up

Assuming that the operation type information is available based on best practice and/or existing manufacturing resources, a tolerance assignment plan dictates how accurate each process should be and what range of the process error and locator error can ensure that accuracy. To comply with industrial standards, the international tolerance (IT) grade is utilized to depict the accuracy level of each process. In the ISO standard, international tolerance grades are numbers which for a particular IT number have the same relative level of accuracy but vary depending upon the nominal or basic size. There are 18 defined tolerance grade bands for each size group. Smaller grade numbers indicate

smaller tolerance zones, and hence more accurate. Table 5.1 and 5.2 illustrate IT grades and corresponding tolerance zones for typical processes.

Different from continuous tolerance-cost function, the IT grades characterize tolerance-cost discretely and give more realistic representation of industrial practices. It is also recognized that IT grades of features produced by the same type of processes vary with machining parameters, operators' skills, machine and tool conditions, fixturing plan, etc. Generally, there is a known or estimated IT grade range associated with each type of manufacturing process. This IT grade range describes the process variability. Any tolerance requirement tighter than the lower limit of this range cannot be achieved by corresponding process. On the other hand, it is not cost effective to use this process for any tolerance requirement looser than the upper limit of the IT grade range.

IT Grade	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Lapping	X	X	X	X											
Honing		X	X	X											
Superfinishing			X	X	X										
Cylindrical grinding			X	X	X	X									
Plane grinding, Broaching, Reaming				X	X	X	X	X							
Boring, Turning					X	X	X	X	X	X	X				
Milling								X	X	X	X	X			
Shaping, Cold Rolling, Drawing									X	X	X	X	X		
Drilling										X	X	X	X		
Die Casting											X	X	X	X	
Forging												X	X	X	X
Sand Casting Hot rolling, Flame cutting													X	X	X

Table 5.1 Machining process associated with ISO tolerance grade

	Nominal Sizes (mm)										
over	1	3	6	10	18	30	50	80	120	180	250
inc.	3	6	10	18	30	50	80	120	180	250	315
IT	Tolerance zone in μm										
1	0.8	1	1	1.2	1.5	1.5	2	2.5	3.5	4.5	6
2	1.2	1.5	1.5	2	2.5	2.5	3	4	5	7	8
3	2	2.5	2.5	3	4	4	5	6	8	10	12
4	3	4	4	5	6	7	8	10	12	14	16
5	4	5	6	8	9	11	13	15	18	20	23
6	6	8	9	11	13	16	19	22	25	29	32
7	10	12	15	18	21	25	30	35	40	46	52
8	14	18	22	27	33	39	46	54	63	72	81
9	25	30	36	43	52	62	74	87	100	115	130
10	40	48	58	70	84	100	120	140	160	185	210
11	60	75	90	110	130	160	190	220	250	290	320
12	100	120	150	180	210	250	300	350	400	460	520
13	140	180	220	270	330	390	460	540	630	720	810
14	250	300	360	430	520	620	740	870	1000	1150	1300
……	……	……	……	……	……	……	……	……	……	……	……

Table 5.2 ISO tolerance zones

5.2 Problem Definition

Suppose n features, q parts, and r machines are involved in a production plan. Manufacturing cost C is formulated as function of assigned tolerance IT grades (IT_i , $i = 1,2,..., n$) and parameters such as features (F_i , $i = 1,2,..., n$), parts (P_j , $j = 1,2,..., q$), machines (M_k , $k = 1,2,..., r$). The goal of tolerance assignment optimization is to minimize manufacturing cost subject to the constraints posed by process capability and design requirements. This constraint optimization problem is then formulated as:

Minimize $C = f\left(IT_i, F_i, P_j, M_k\right), i = 1,2,..., n;\ j = 1,2,..., q;\ k = 1,2,..., r$

s.t.: 1) $\min\left[IT\left(F_i\right)\right] \le IT_i \le \max\left[IT\left(F_i\right)\right]$, and

2) $Tol_{SIM} \le Tol_{REQ}$ for all design tolerances

$IT\left(F_i\right)$ is feasible IT for the ith feature; Tol_{SIM} is the stack up simulation results; Tol_{REQ} is the design tolerance requirement.

Specifically for a tolerance assignment task, however, other parameters such as features ($F_i, i = 1,2,..., n$), parts ($P_j,\ j = 1,2,..., q$), machines ($M_k,\ k = 1,2,..., r$), are treated as known constants. Tolerance IT grades ($IT_i, i = 1,2,..., n$) become the only set of variables. Thus, the objective C is reduced to: $C = f\left(IT_i\right),\ i = 1,2,..., n$

5.3 Cost Model

In [52], the complete cost model consists of models on machine level, part level, and feature level. Here, a simplified model is adopted, which only takes into consideration the influence of the IT grades on feature level.

At the feature level, the cost depends on material machinability, feature type, size, and IT grade. With material, feature type and size as known factors retrieved from design information, the IT grade is the only variable at this level.

$$C\left(F_i\right) = \alpha \cdot \beta \cdot f_1\left(F_i\right) \cdot V\left(F_i\right) \cdot \exp\left(a \cdot IT_i\right) \qquad i = 1,2,..., n$$

$C(F_i)$ is the manufacturing cost of the *i*th feature; α is the material machinability factor; β is feature complexity factor; $f_1(F_i)$ is the cost factor associated with type of the *i*th feature; $V(F_i)$ is the volume of material to be removed in order to produce the *i*th feature; *a* is a constant to be determined. The cost factors for different feature types can be estimated according to previous manufacturing practice or existing cost data. Table 1 shows some cost factor examples. The feature complexity factor is introduced because the same type of features may result in different manufacturing cost due to different complexity. For example, a long, narrow hole is more costly compare with a short, broad hole even they have same volume of unwanted material and assigned IT grades. With this formulation, the manufacturing cost of any known single feature can be determined.

Feature type	Cost factor	Feature type	Cost factor
Flat surface	1	External thread	1.75
hole	1	T slot	2
block slot	1	Internal spline	2
chamfer	1	Y slot	2.25
radial groove	1.25	External spline	2.25
Keyway	1.5	Internal thread	2.25
V slot	1.5	face groove	2.5

Table 5.3 Manufacturing cost factors for different feature type

5.4 Tolerance Assignment Using GA

5.4.1 Encoding

In a given series of processes, suppose there are totally n processes, then there are *n* corresponding tolerances that need to be determined with a IT grade. Thus, a candidate tolerance assignment plan can then be represented as a chromosome in such a form: X={IT(1), IT(2), ……, IT(n)}, where IT(i)$\in${possible IT grades for this type of process} (i=1,2,…n). For example, suppose the *i*th process is plane grinding. Then the possible IT grade range is from IT grade 5 to 9.

5.4.2 Objective Function Definition

Objective function should represent the both aspects discussed in problem definition section: 1) the goal of minimizing manufacturing cost; and 2) design requirement constraints are satisfied. Therefore, objective function should consists of two terms correspondingly:

$$F(X)=C(X)+P(X)$$

F(X) is the objective function. C(X) is the manufacturing cost function. P(X) is a penalty function that penalize those tolerance assignment plans that fail to meet the design requirements.

$$P(X)= const * IND(X)$$

const is a weight coefficient, and *IND(X)* is an indicator function.

$$IND(X) = \begin{cases} 0, & \text{if the tolerance assignment plan X passes the design requirements} \\ 1, & \text{if the tolerance assignment plan X fails the design requirements} \end{cases}$$

The goal is to minimize the value of objective function. The value of *const* should be large enough comparing to the cost function so as to penalize failed plans by significantly increase the value of their objective function. By defining objective function as a summation of cost and penalty function, the original constraint optimization problem turns into a non-constraint optimization problem.

5.4.3 GA Implementation

Standard GA procedures are carried out:

1. Initialize a population of chromosomes representing tolerance assignment plans;
2. Evaluate each chromosome using the objective function;
3. Allocate numbers of copies for each chromosome to form the intermediate generation (select)
4. Group chromosomes in the intermediate generation and switch part of their genes (Crossover)
5. Choose genes at a random location and alter its value. (Mutation)
6. Repeat steps 2~5 until the number of loops reaches a certain amount or satisfied plans are achieved

GA Parameters and types of reproduction operators need to be specified at the beginning, such as population size, initialization method, selection method, crossover method, crossover rate, mutation rate, etc.

5.5 Case Study

5.5.1 Workpiece and Processes

The workpiece being machined and all its design requirement is shown in Figure 5.2. To machine all the features, there are totally twelve processes. The process information is shown in Table 5.4.

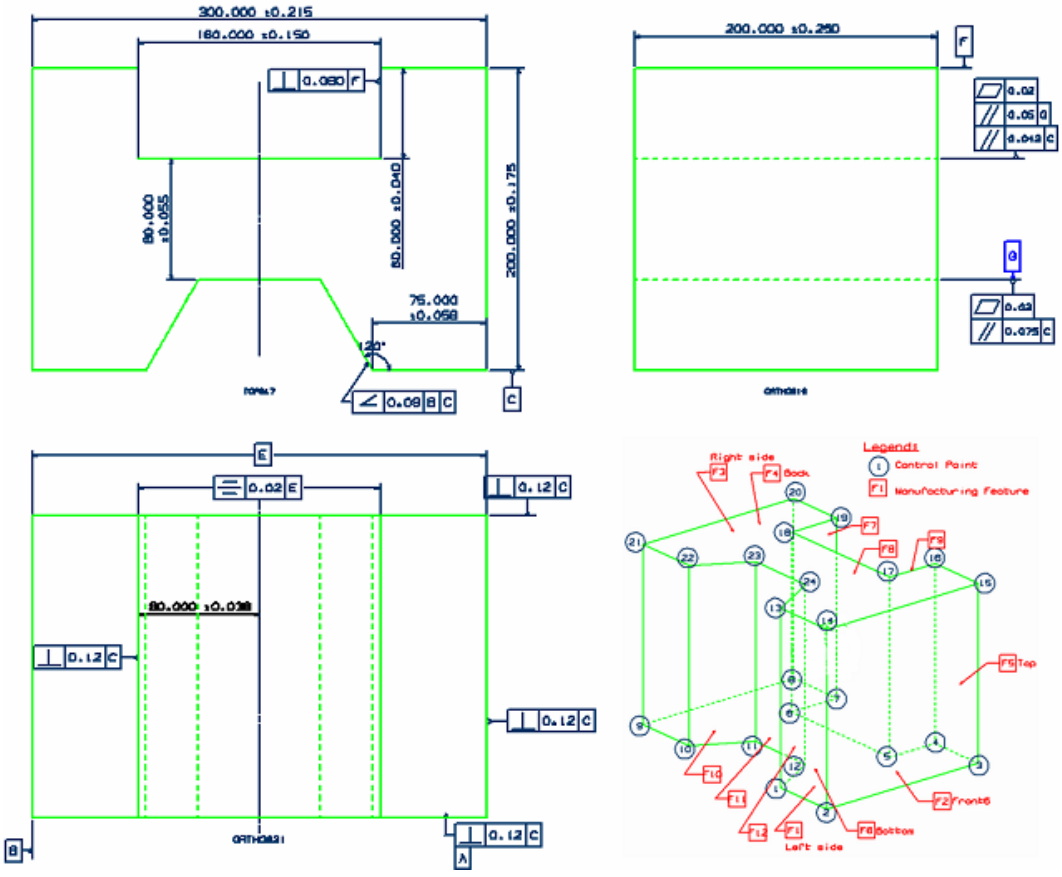

Figure 5.2 Sample workpiece and design requirements

SETUP	PROCESS	FEATURE	MACHINING TYPE
SETUP I: Locating Surface: 6,3,4	1	Plane 2	Face mill
	2	Plane 5	Face mill
SETUP II: Locating Surface: 5,3,2	3	Plane 3	Face mill
	4	Plane 6	Face mill
	5	Plane 10	Profile mill
	6	Plane 12	Profile mill
	7	Plane 11	End mill
SETUP III: Locating Surface: 2,5,3	8	Plane 1	Face mill
	9	Plane 4	Face mill
SETUP IV: Locating Surface: 6,3,4	10	Plane 7	Slot mill
	11	Plane 9	Slot mill
	12	Plane 8	End mill

Table 5.4 Process information

5.5.2 GA-based Tolerance Assignment

There are totally twelve processes in this case, so there are twelve IT grades to be determined. For face milling, profile milling, end milling and slot milling, the IT grades all range from IT grade 5 to 9. Thus a chromosome representing a tolerance assignment plan is X={IT(1), IT(2), ……, IT(12)}, where IT(i)$\in${5, 6, 7, 8, 9} (i=1,2,…n).

The objective function is defined as:

$$F = \underbrace{C + P = \sum_{i=1}^{12} \exp(-0.1 \times IT(i) + 1) \times Size(i) \times A(i)}_{Cost} + \underbrace{const \times IND(X)}_{\substack{Constra\,int \\ Penalty}}$$

where *Size* = {4, 6, 6, 6, 1.38564, 1.3856, 1.6144, 6, 4, 1.2, 1.2, 3.2}, and *A* = {0.855, 0.855, 0.855, 0.81, 0.99, 0.81, 0.99, 0.855, 0.855, 0.81, 0.81, 0.855}. *const* = 100.

The initial population is generated on a random basis. Population size is 50. The evolution process adopts proportional selection and two-point crossover. Crossover rate is 0.6, and mutation rate is 0.001. The evolution terminates after 20th generation.

5.5.3 Simulation Results and Analysis

The simulation is performed based on a GA simulation package—Genesis 1.0. The original code was modified to tailor to the application of this case, including the encoding method and objective function definition. Within 20 generations, the optimal tolerance assignment plan we get is listed in Table 5.5.

IT1	IT2	IT3	IT4	IT5	IT6	IT7	IT8	IT9	IT10	IT11	IT12	Cost Func.
9	6	9	8	7	6	8	9	9	8	5	9	43.78

Table 5.5 Optimal tolerance assignment plan achieved by GA

Tolerance assignment task was operated on the same workpiece and process design using sensitivity analysis [52], and the optimal plan is listed in Table 5.6.

IT1	IT2	IT3	IT4	IT5	IT6	IT7	IT8	IT9	IT10	IT11	IT12	Cost Func.
9	5	9	9	6	6	6	9	9	7	6	7	45.29

Table 5.6 Optimal tolerance assignment plan achieved by sensitivity analysis

The optimal tolerance assignment plans achieved by both methods pass the design requirements. However, within comparable computation time, the plan achieved by GA has smaller objective function value, which means GA-based method provides looser overall operation tolerance plan, hence is better in terms of lower manufacturing cost.

Now we look further into the data associated with GA evolution process. The manufacturing cost of the best tolerance assignment plan in each generation is recorded through generations. The cost generally decreases over the number of generations, as shown in Figures 5.3, which demonstrates GA's effect of continuous improvement.

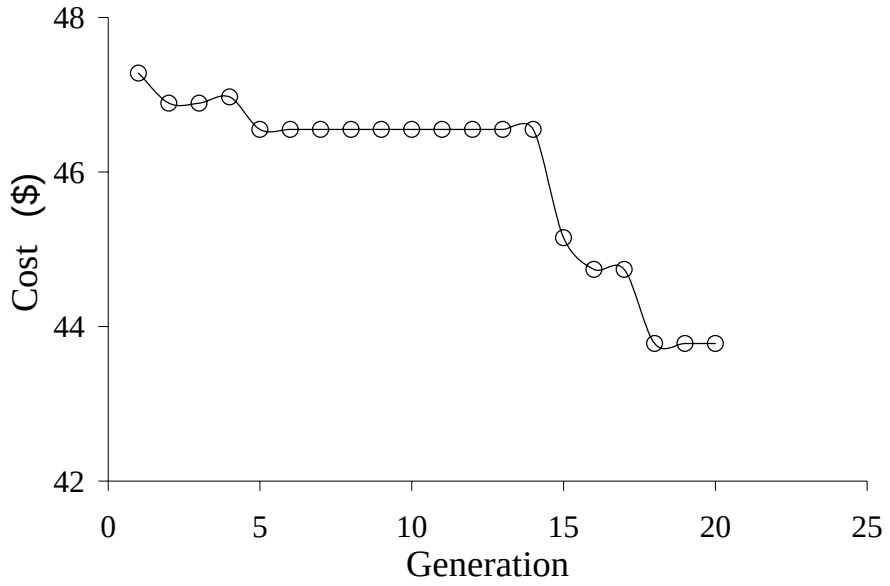

Figure 5.3 Cost of the best tolerance assignment plan in each generation

More information can be retrieved regarding the evolution pattern of IT grades for a single process. Figure 5.4 shows that over generations, IT grades for some processes tend to increase while some tend to decrease, and some goes up and down. Processes with wither increasing and decreasing IT grades are processes more dominant to the performance of a candidate plan. On the contrary, the processes for which the IT grades do not have a uniform evolving pattern are less important. A clear increase in IT grades indicates that the tolerance for this process should definitely be loosened in order to lower cost. On the other hand, a decreasing means this process is critical and needs extra accuracy during operation.

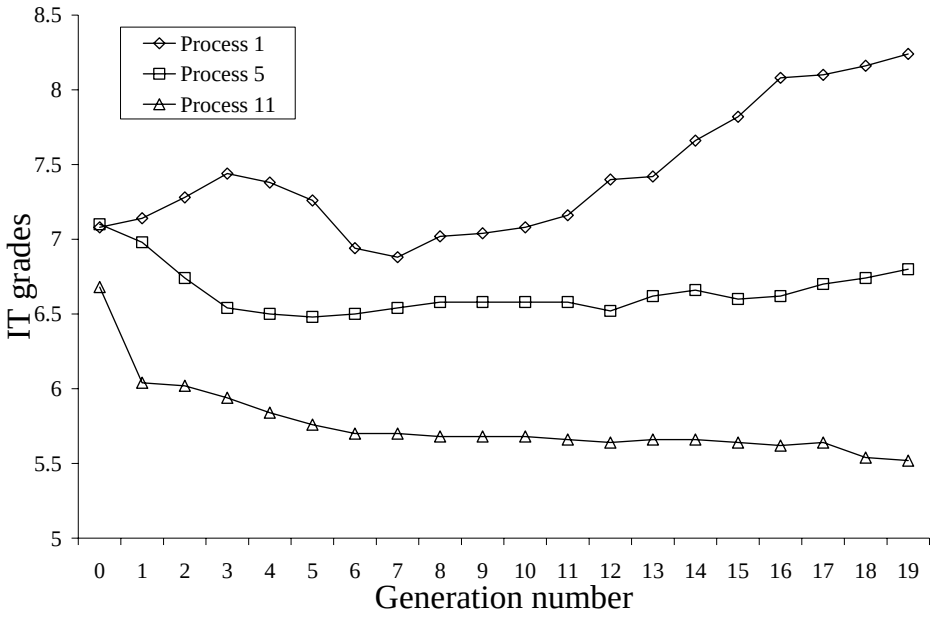

Figure 5.4 Evolution of IT grades in selected processes

In Figure 5.5, average IT values of each process to form a complete synthesis plan are plotted for selected generation. At generation 0, IT values for each process are randomly generated and hence has not pattern. When it reached generation 4, the GA

already picked up the trend of evolution of each process and demonstrated an immature pattern. At generation 19, the overall fitness of candidates is approaching the state of saturization to provide near optimal solution for tolerance synthesis.

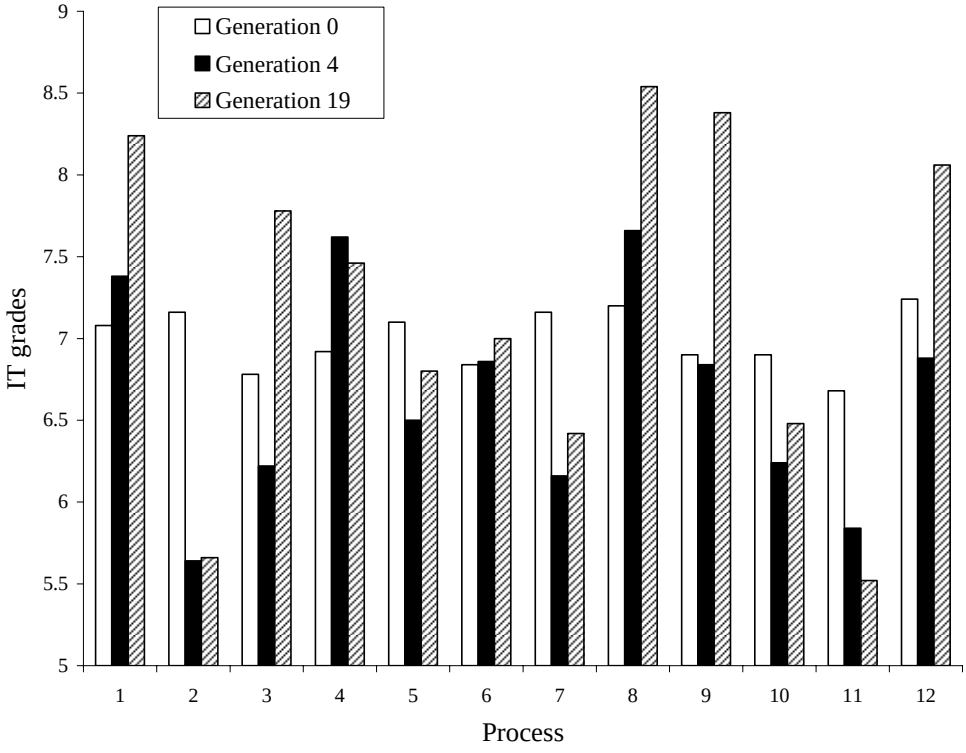

Figure 5.5 IT grades at selected generations

5.6 Summary

Due to the large solution space, multi-process tolerance assignment problem is usually hard to solve using traditional optimization methods. Genetic Algorithm has shown its power as an optimization tool in this problem.

The multi-process tolerance assignment problem was formulated. The methodology of using Genetic Algorithms to optimize tolerance assignment plan was

systematically presented. In a case study, the results from Genetic Algorithm showed to be better than those of sensitivity analysis. Also, further analysis of the data during GA implementation provides another possible way to better understanding the degree of influence of the tolerances in different processes. In a word, GA has proved to be a capable tool in computer-aided tolerance assignment, and is helpful in the reduction manufacturing cost through better process design.

CHAPTER 6 CONCLUSION AND FUTURE WORK

In this thesis, both theoretical and application aspects of GA are studied. On the theoretical side, an improved GA—Evolution Direction Guided-GA (EDG-GA) is developed to enhance the computation efficiency. Also, a method is proposed to study the structure of GA solution space. On the application side, GA is used to solve the tolerance assignment problem in manufacturing.

The design of EDG-GA is based on the analysis of Schema Theorem and Building Block Hypothesis. Large increase in the number of occurrence of a certain allele loosely indicates the allele is likely to be good. With this understanding, a new genetic operator is designed to create a new chromosome in each generation. The new chromosome combines all the alleles with the largest increase in their number of presence from last generation. As evolution goes on, this chromosome is essentially a real-time projection of potentially fit chromosome. By adding such an adaptive chromosome to the population, the selection bias is strengthened and evolution is directed towards potentially fit direction. This new genetic operator is effective in improving the computation efficiency of GA when the interaction between genes is not significant.

Centered with the issue of gene interaction, a method is proposed to study the structure of solution space. Gene-Chromosome Correlation Function is first defined to characterize the contribution of each individual gene to the chromosome fitness. Based on this function, dependency matrix is defined to address the interactions between genes. Dependency matrix represents the structure of solution space by recording the degree of

interactions between every two genes. The modified dependency matrix is further used to determine crossover points. Comparing to random crossover, selective crossover identifies and preserves building blocks, which will make GA more efficient.

GA is used for manufacturing tolerance assignment problem. GA is proved to be successful in designing optimal tolerance assignment plan for multi-setup/multi-process operations. Designed with GA, the final tolerance assignment plan incurs lower manufacturing cost.

Although this study improves the performance of GA under some circumstances, limitations still exists. As discussed earlier, EDG-GA's effectiveness will be restrained for problems with significant interdependence between variables. This can be somewhat ameliorated if we combine EDG-GA with the methods presented to address gene interactions. However, details of the solution need to be developed.

Conceptually, the logic base of the proposed algorithm and method is relatively sound. However, due to the uncertainty nature of meta-heuristic algorithms, the effectiveness of the algorithm and method needs further validation through more case studies.

The successful application of GA in tolerance assignment, together with that in many other areas, suggests the usefulness of GA in handling large scale design optimization problems. On one hand, GA can be applied to solve complicated problems in additional fields of study, such as materials design, scheduling, image analysis, etc. On the other, GA's power can be extended beyond the class of traditional design optimization problems. For instance, GA can be used in knowledge acquisition from

mass data. To use GA in these alternative types of problems, the formulation of the physical problem in GA's perspective will be the first issue and one of the most important ones.

REFERENCES

1. S. Brocchini, K. James, V. Tangpasuthadol and J. Kohn, "A Combinatorial Approach for Polymer Design", J. of American Chemistry Society, Vol. 119, No. 19, 1997, pp. 4553-4554

2. G. Vignaux and Z. Michalewicz, “A Genetic Algorithm for the Linear Transportaion Problem”, IEEE Trans. Systems, Man and Cybernetics, Vol. 21, No. 2, 1991, pp. 445-452.

3. S. Lin, “Genetic Algorithm-Based Scheduling System for Dynamic Job Shop Scheduling Problem”, Ph.D Dissertation, Michigan State University, 1997.

4. L. Wang, Intelligent Optimization Algorithms with Applications, Tsinghua University Press, Beijing, China, 2001. (in Chinese)

5. I. Osman and G. Laporte, “Metaheuristics: A Bibliography” Annals of Operations Research, Vol. 63, 1996, pp. 513–623.

6. C. Blum and A. Roli, “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison” ACM Computing Surveys, Vol. 35, No. 3, 2003, pp. 268–308.

7. N. Metropolis et al., “Equations of State Calculations by Fast Computing Machines”, J. of Chemical Physics, Vol. 21, 1953, pp. 1087-1091.

8. F. Glover, “Future Paths for Integer Programming and Links to Artificial Intelligence", Computers and Operations Research, Vol. 13, 1986. pp. 533-549.

9. J. Hopfield, “Neural Networks and Physical Systems with Emergent Collective Computational Abilities” in: Proc. of National Acadamy of Science, Vol 81, 1982, pp. 3088-3092.

10. J. Holland, Adaption in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

11. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

12. S. Skiena, The Algorithm Design Manual, Spring-Verlag, New York, NY, 1998.

13. L. Howard and D. D'Angelo, “The GA-P: A Genetic Algorithm and Genetic Programming Hybrid”, IEEE Expert, Vol. 10, No. 3, 1995, pp.11-15.

14. L. Yao, “Parameter Estimation for Nonlinear Systems”, Ph.D Dissertation, University of Wisconsin-Madison, 1992.

15. J. Joines, C. Culbreth and R. King, “Manufacturing Cell Design: An Integer Programming Model Employing Genetics”, IIE Trans., Vol. 28, No. 1, 1996, pp. 69-85.

16. F. Gold, “Application of the Genetic Algorithm to the Kinematic Design of Turbine Blade Fixtures”, Ph.D Dissertation, Worcester Polytechnic Institute, 1998.

17. S. Timothy, “Optimization of Sequencing Problems using Genetic Algorithms”, Ph.D Dissertation, Colorado State University, 1993.

18. J. Davern, “Architecture for Job Shop Scheduling with Genetic Algorithms”, Ph.D Dissertation, University of Central Florida, 1994.

19. S. Rho, “Distributed Database Design: Allocation of Data and Operations to Nodes in Distributed Database Systems”, Ph.D Dissertation, University of Minnesota, 1995.

20. S. Tadikonda, “Automated Image Segmentation and Interpretation using Genetic Algorithms and Sensing Nets”, Ph.D Dissertation, University of Iowa, 1993.

21. R. Huang, “Detection Strategies for Face Recognition using Learning and Evolution”, Ph.D Dissertation, George Mason University, 1998.

22. C. Reeves and J. Rowe, Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory, Kluwer, Boston, MA, 2003.

23. C. Stephens and H. Waelbroeck, “Schemata Evolution and Building Blocks”, Evolutionary Computation, Vol. 7, No. 2, 1999, pp. 109-124.

24. A. Nix and M. Vose, “Modeling Genetics with Markov Chains”, Annals of Mathematics and Artificial Intelligence, Vol 5, 1992, pp.79-88

25. D. Goldberg, Lessons from and for Competent Genetic Algorithms, Kluwer, Boston, MA, 2002.

26. D. Goldberg, B. Korb and K, Deb “Messy Genetic Algorithms: Motivation, Analysis and First Results”, Complex Systems, Vol. 3, 1989, pp. 493-530.

27. G. Harik and D. Goldberg, “Learning Linkage”, Foundations of Genetic Algorithms, Vol. 4, 1997, pp. 247-262.

28. M. Pelikan and K. Sastry, “Fitness Inheritance in the Bayesian Optimization Algorithm”, in: Proc. of Genetic and Evolutionary Computation Conference, Vol. 2, 2004, pp. 48-59.

29. Y. Davidor, “Epistasis Variance: Suirability of A Representation to Genetic Algorithms”, Complex Systems, Vol 4, 1990, pp. 369-383.

30. D. Beasley, D. Bull and R. Martin, “Reducing Epistasis in Combinatorial Problems by Expansive Coding”, in: Proc. of 5th Intl. Conf. on Genetic Algorithms, 1993, pp. 400-407.

31. E. Cantu-Paz, “Efficient and Accurate Parallel Genetic Algorithms”, Kluwer, Boston, MA, 2000.

32. A. Bethke, “Comparison of Genetic Algorithms and Gradient-Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity”, Tech. Rep. No. 197, Logic of Computers Group, University of Michigan, 1976.

33. P. Grosso, “Computer Simulations of Genetic Adaption: Parallel Subcompnent Interaction in a Multilocus Model”, Ph.D Dissertation, University of Michigan, 1985.

34. A. Sinha, “Designing Efficent Genetic and Evolutionary Algorithm Hybrids”, M.S. Thesis, University of Illinois-Urbana-Champaign, 2002.

35. R. Srivastava, “Time Continuation in Genetic Algorithms” M.S. Thesis, University of Illinois-Urbana-Champaign, 2002.

36. Y. Jin, M. Olhofer and B. Sendhoff, “On evolutionary Optimization with Approximate Fitness Functions”, in: Proc. of Genetic and Evolutionary Computation Conference, 2000, pp. 786-793.

37. R. Smith, B. Dike and S. Stegmann, “Fitness Inheritance in Genetic Algorithms”, in: Proc. of the ACM Symposium on Applied Computing, 1995, pp. 345-350.

38. B. Ngoi, and C. Ong, "Product and Process Dimensioning and Tolerancing Techniques: A State-of-the-Art Review", Intl. J. of Advanced Manufacturing Technology, 1998, pp. 910-917.

39. P. Ostwald and J. Huang, "A Method for Optimal Tolerance Selection", J of Engineering for Industry, 1977, pp. 558-564.

40. X. Tang and B. Davies, "Computer Aided Dimensional Planning", Intl. J. of Production Research, 1988, pp. 283-297.

41. M. Kalajdzic, D. Domazet and S, Lu, "Current Design and Process Planning of Rotational Parts", Annals of CIRP, 1992, pp. 181-184.

42. S. Manivannan, A. Lehtihet and P. Egbelu, "A Knowledge Based System for the Specification of Manufacturing Tolerances", J. of Manufacturing Systems, 1989, pp. 153-160.

43. A. Shan and R. Roth, "Genetic Algorithms in Statistical Tolerancing ", Mathematical and Computer Modeling, Vol. 38, 2003, pp. 1427-1436.

44. S. Lu, and R. Wilhelm, "Automating Tolerance Synthesis: A Framework and Tools", J. of Manufacturing Systems, 1989. pp. 279-296.

45. C. Ouyang, T. Lin, "Developing an Integrated Framework for Feature-Based Early Manufacturing Cost Estimation", Intl. J. of Advanced Manufacturing Technology, 1997, pp. 618-629.

46. M. Mayer and M. Nusswald, "Improving Manufacturing Costs and Lead Times with Quality-Oriented Operating Curves", J. of Materials Processing Technology, 2001, pp. 83-89.

47. A. Esawi and M. Ashby, "Cost Estimates to Guide Pre-selection of Processes", Materials and Design, 2003, pp. 605-616.

48. J. Feng, A. Kusiak and C. Huang, "Cost Evaluation in Design with Form Features", Computer-Aided Design, 1996, pp. 879-885.

49. I. Weustink, E. Brinke, A. Streppel and H. Kals, "A Generic Framework for Cost Estimation and Cost Control in Product Design", J. of Materials Processing Technology, 2000, pp. 141-148.

50. K. De Jong, "An Analysis of the Behavior of A Class of Genetic Adaptive Systems", Ph.D Dissertation, University of Michigan, 1975.

51. A. Yassine, D. Falkenburg and K. Chelst, "Engineering Design Management: An Information Structure Approach", Intl. J. of Production Research, Vol. 37, No. 13, 1999, pp. 2957-2975.

52. H. Song, Y. Yang, Y. Zhou, and Y. Rong, “Tolerance Assignment using Genetic Algorithm Optimization for Production Planning”, The 9th CIRP International Seminar on Computer-aided Tolerancing, Tempe, AZ, Apr 10-12, 2005.